Position Paper Draft http://ti.arc.nasa.gov/sc05/
20th IEEE/ACM International Conference on Automated Software Engineering
Software Certificate Management Workshop, SoftCeMent05
November 8, 2005
Long Beach, California, USA

Qualification of Software Development Tools for Airborne Systems Certification

Andrew J. Kornecki,
Embry Riddle Aeronautical University, Daytona Beach, FL
Janusz Zalewski
Florida Gulf Coast University, Fort Myers, FL

Introduction - Design Tools Categorization

This position paper is a result of recent research contracted by the Federal Aviation Administration. The main objective of this research was to identify the assessment criteria that allow both developers and certifying authorities to evaluate specific safety critical real-time software development tools from the system/software safety perspective. Related objectives were: to present and evaluate the state of the art in safety critical software development tools and to establish a basis for software development tool qualification guidelines. The research included literature reviews and industry surveys, creation of tool evaluation taxonomy, installation of and experiments with the selected tools and collecting and analyzing data. The focus of this research was on real-time design tools with automatic code generation features.

During the design of a real-time system, it is important to be aware that there exist two distinct classes of modern systems exposed to environmental stimuli:

- Interactive – the computer system determines the pace of operation by granting or allocating resources to clients on request when feasible (operating systems, data bases); the concerns are deadlock avoidance, fairness, data coherence.
- Reactive – the system environment determines the pace of operation while the computer system reacts to external stimuli producing outputs in timely way (process control, avionics, signal processing); the concerns are correctness and timeliness.

Consequently, design tools that assist developers in translating the requirements into source code can be categorized into two groups, selected for this study:

- function-based, block-oriented, reactive, preferred by engineers with system and control background, and
- structure-based, object-oriented, interactive, preferred by engineers with software and computer background.

Current Practice of Development Tools Qualification

The existing software aspects of airborne systems certification guidelines defined by the FAA through Advisory Circular 20-115B [1] and DO-178B [2] define software development tools as: *"Tools whose output is part of airborne software and thus can introduce errors"*. As elaborated in DO-178B and related documents [3,4], qualification is a supplementary process the applicant may elect to follow in a course of certification for the airborne system. Tool qualification is attempted only as an integral component of a specific certification program, i.e., part of a Type Certificate (TC), Supplemental Type Certificate (STC), or Technical Standard Order (TSO) approval. The tool data are referenced within the Plan for Software Aspects of Certification (PSAC) and Software Accomplishment Summary (SAS) documents for the original certification project. The applicant should present for review the Tool Operational Requirements (TOR) – a document describing tool functionality, environment, installation, operation manual, development process, expected responses (also in abnormal conditions). Two other documents must be submitted: Tool Qualification Plan and Tool Accomplishment Summary. To make an argument for qualification, the applicant must demonstrate that the tool complies with its TOR. This demonstration may involve a trial period during which a verification of the tool output is performed and tool-related problems are analyzed, recorded and corrected. The document states also that software development tools should be verified to check the correctness, consistency, and completeness of the TOR and to verify the tool against those requirements. More data is required for the qualification of a development tool, including Tool Configuration Management Index, Tool Accomplishment Summary, Tool Development Data, Tool Verification Data, Tool Quality Assurance Records, Tool Configuration Management Records, etc. These requirements are described in chapter 9 of the FAA Order 8110.49 [4].

Currently, there is no central repository that maintains a listing of previously qualified tools. Only the applicant, who qualified a tool within the scope of a specific project, has or owns the necessary data. The conducted survey identified only a handful of development tools that have been qualified. In addition, the obtained information is anecdotal based on personal contacts and word of mouth rather than documented in a way that could be examined in detail. The only information the research team was able to obtain were statements to the effect: "Tool X was qualified while used on Level-Y certified system Z by applicant W". Occasionally, additional information was conveyed in a form of brief e-mails discussing the qualification approach or other details. More often the follow-up was unanswered. Efforts to get an access to specific documentation were unsuccessful. We may hypothesize that main reasons for that are intellectual property rights, company policies guarding competitive advantage, and the business red tape.

One needs to note that modern commercial development tools are typically complex suites combining multiple functionalities/capabilities. The survey identified a relatively short list of qualified development tools, or more specifically: selected functionality of the tool suite. At the time of this writing they included code generators (GALA: Generation Automatique de Logiciel Avionique, GPU: Graphical Processing Utility, VAPS CG: Virtual Application Prototyping System Code Generator, SCADE QCG: Safety Critical Avionic Development Environment Qualifiable Code Generator) and configuration-scheduling table generators (UTBT: Universal Table Builder Tool, CTGT: Configuration Table Generation Tool), most of them being in-house products.

Assessment Criteria and Experiments

In order to help in the tool qualification process, we first attempted to identify criteria (metrics) for tool evaluation, establish their measures, and conduct experiments validating the approach. Of the multiple criteria that have been used for tool evaluation in the past, we selected the following four [5], listed here with their measures:

- *Usability* measured as development effort (in hours)
- *Functionality* measured via the questionnaire (on a 0-5 points scale)
- *Efficiency* measured as code size (in LOC)
- *Traceability* measured by manual tracking (in number of defects).

This selection was based on the analysis of documentation from the FAA, ISO/IEC standards, industry survey, and studies done by the British Computer Society and Technical Center of Finland.

The experiments that followed were conducted on a generic avionics system test-bed equipped with a flight simulator, with real industry-strength software design tools [5]. Data were collected in four phases of the design process including:

(1) Project preparation

(2) Model creation and code generation

(3) Actual measurement

(4) Postmortem.

The recently published results [6] confirm that this approach can be effectively used to distinguish between tools within the group of established criteria.

Selected Additional Observations

- Despite their diversity, complex design tools with analysis and code generation capability dominate software tools market. Software for civil aviation systems is risk averse and provides a low quantity market not having enough commercial clout to drive the software tool market. On the other hand, use of tool on specific safety-critical project is a good public relation opportunity for tool vendor, which may result in increase of sales in less-regulated industries.
- Modern complex multifunctional tools require a steep learning curve. Considering tool complexity, the quality of support materials is often marginal. Unless developers become expertly proficient with the tool, reliance on it may lead to ignorance of tool functionality and complacency.
- The notation used by specific tool constraints the design options, thus restricting design flexibility. The tools may exhibit behavioral discrepancy due to the underlining runtime model.
- No mechanism exists to promulgate information about tool qualification. The qualification data constitute component of the certification package and are highly proprietary.
- It is imperative that the objectives for development tool qualification reflect the fact that the modern tools operate in an environment different than the target system. The typical operating environment for a tool is a general-purpose COTS workstation under conventional operating system. The critical issue for the tools is the integrity of the data as opposed to the tool operation in terms of timing, memory use, etc.
- Service history provides a means for claiming partial certification credits for target software. It does not help greatly to provide means for the development tools qualification, due to rapid progress of software technology. Typically, by the time

enough data is collected to create appropriate service history, the tool has been updated or modified in some way. Thus, in general, there is not enough time to get service history data for a development tool.

Conclusion

The experiments and observations indicate that the industry might benefit from methods to qualify a tool that are independent of a specific program and the applications using it. This would require updating the guidelines to consider that the tools operate in ground-based COTS environment different from the target application. This would also require considering a model-based development paradigm, re-defining the qualification process and allowing flexibility regarding qualification that is less dependent on the application program using the tool. A service history approach, considering incremental tool changes, may be also needed. A more streamlined method to qualify development tools and to keep them current as technology advances would be also useful. The streamlining must, however, not compromise safety.

References

[1] U.S. Dept. of Transportation, Federal Aviation Administration, *Advisory Circular AC 20-115B*, November 1993

[2] Radio Technical Commission for Aeronautics, *Software Considerations in Airborne Systems and Equipment Certification*, RTCA DO-178B, RTCA SC-167, 1992

[3] Radio Technical Commission for Aeronautics, *Final Report for Clarification of DO-178B 'Software Considerations in Airborne Systems and Equipment Certification'*, RTCA DO-248B, RTCA SC-190, 2001

[4] U.S. Department of Transportation, Federal Aviation Administration, *Software Approval Guidelines, FAA Order 8110.49, (Chapter 9 replaces FAA Notice N8110.91 of 2001)*, FAA, 2003

[5] Kornecki A., Zalewski J., *Process Based Experiment for Design Tool Assessment in Real-Time Safety-Critical Software Development*, Proc. SEW-29, Annual 29th NASA/IEEE Software Engineering Workshop, Greenbelt, MD, April 6-7, 2005

[6] Kornecki A., J. Zalewski, *Experimental Evaluation of Software development Tools for Safety-Critical Real-Time Systems*, Innovations in Systems and Sofware Engineering – A NASA Journal, Vol. 1, No. 2, October 2005